

Managing Large HW/SW Codesign Projects

Carsten Wolff¹, Idania Gorrochategui², Markus Bucker³

¹: University of Applied Sciences and Arts Dortmund
Emil-Figge-Str. 42, 44227 Dortmund, Germany
carsten.wolff@fh-dortmund.de

²: University of the Basque Country – Bilbao
Barrio Sarriena s/n, 48940 Leioa, Bizkaia, Spain

³: Intel Mobile Communications GmbH
Düsseldorfer Landstr. 401, 47259 Duisburg, Germany

Abstract—HW/SW Codesign [4] is a strategy to develop highly optimized systems composed of both electronics hardware and software. Examples are smartphones, consumer electronics or embedded systems. The challenge is to achieve an optimized split by partitioning the system components into hardware and software and to achieve the best project execution time (and cost) by developing both hardware and software concurrently [1]. If such systems contain hundreds of highly interdependent hardware and software components, the R&D project for such a system becomes large and very complex. Since more HW/SW systems are developed and since the complexity is growing, a specialized project management methodology for this kind of projects is emerging. Our contribution is focused on giving a practical overview of the challenges and best practices for the management of such projects. The findings are based on more than 10 years industrial experience with various large smartphone and consumer electronics projects. Currently, these findings are evaluated in a research project. This contribution gives an overview on the scope and future targets of this project.

Keywords—R&D Projects, Large Projects, HW/SW Codesign

I. INTRODUCTION

The development of modern electronic consumer devices requires the design of both hardware and software components. It is necessary to do a good partitioning between hardware and software components and to integrate both parts to an optimized overall design. A processor for a mobile phone can consist of several hundred hardware and software blocks, they need to be combined to a consistent functionality. Usually, it is not obvious which components will be done in hardware and which will be done in software, this is subject to multidimensional optimization problems and quite often done based on heuristics [1]. The resulting HW/SW system is highly interdependent. This leads to a concurrent

development paradigm, both parts – hardware and software – need to be developed at the same time to allow an optimization of the partitioning and to validate and verify the quality of the partitioning [1]. Early simulations and rapid prototyping are used to allow a proof of concept. The software components are usually required to verify the functionality of the hardware; therefore, the availability of early software releases is necessary. The software developers need models of the later hardware to verify their components. This is solved by concurrent engineering. Apart from solving the co-design issues, the concurrent approach is required to shorten project cycle times.

Consequently, large HW/SW codesign projects require special treatment in terms of project management, communication and organization [5][14]. Due to their interdependent and concurrent character a methodology for the effective and efficient interaction between the involved sub-teams has to be applied [8]. Our contribution gives findings and recommendations based on project experience. The characteristics and challenges are described briefly based on a typical project case. Some important basic rules for handling the projects are given.

II. PROJECT INTEGRATION

A. Outline of a typical project

A typical project for a mobile phone processor has a runtime of 2 years and involves 100 – 200 software and hardware engineers. These engineers are distributed over 4-6 different sites located in different countries and time zones.

The project starts with a product specification based on requirements engineering. This specification is refined during project runtime and can be adapted based on change requests. Verification and validation have to prove consistency between the processor and the final version of

the specification. The specification contains a first version of the HW/SW split.

Based on the specification code is developed describing both hardware and software modules. To verify the code, both hardware and software are simulated. The software and hardware components are combined in these simulations. A software representation (virtual prototype) can be used to develop the target software at a very early stage of the project. In addition rapid prototyping based on FPGAs (Field Programmable Gate Arrays) can create accelerated hardware prototypes and help with an opportunity for early verification to validate the hardware functionality.

After creating a verified and validated code base for both software and hardware, the hardware part is used to manufacture engineering samples of the processor chip. The samples are used to run the software modules and to prove the functionality of the HW/SW system in a testbed environment. The manufacturing of the samples is pretty expensive (several million EUR) and time consuming (> 10 weeks). Therefore, it is insisted to avoid iteration cycles for bug-fixing. The target is a first-time-right design, meaning, the design and verification phase before manufacturing the samples has to be very careful.

B. Main Challenges

The highly interdependent character of the project requires frequent exchange of releases of the hardware modules to the software developers and vice versa. In addition, the product specification has to take new versions of the hardware and software into account. New hardware and software components have to be integrated into the complete design to allow simulation and verification. In conclusion, the complete HW/SW systems have to be updated with new block releases and integrated to a consistent overall design almost on daily basis. This means that 10-15 different sub-teams have to deliver defined versions of their components according to a tight schedule. A late or buggy release of a component may lead to an inconsistent, delayed or buggy overall design and can stop all other teams from moving on with their components. Furthermore, for the integration work for all the different modules, it has to be clear which features and bug fixes are included in a component release. A hardware component that does not fit to the respective software component may lead to a wrong overall system.

Due to this concept, it is useful to develop software on base of a virtual prototype, which is also used as a reference for hardware simulation. By doing this, a full parallel development of hardware and software components is feasible even allowing the software to be ready before the hardware.

The crucial factor for this process is to ensure that the hardware abstraction models used for software development are used as well to validate the hardware.

III. BEST PRACTICES

For the fast and reliable execution of large and complex HW/SW codesign projects an elaborated strategy has emerged over the past ten years. Project planning reflects the concurrent development paradigm instead of a traditional sequential execution. Nevertheless, a highly interconnected project plan with short cycle times and few buffers tend to be delayed and disturbed by any small issue in one of the many tasks. Such a plan is sensitive even to small changes [2]. Therefore, a very strict discipline between all involved parties is mandatory. The plan itself has to be designed in a way that allows frequent updates and a good overview on how sensitive and critical the whole project is towards issues and delays.

A. Sub-Project Organization

One recipe for handling such projects lies in the right project organization. Since the project is too large to be done in one team and since the hierarchical composition of the resulting product from various hardware and software components is obvious, a project organization with several sub-teams is recommended [7][12]. These teams can be set up according to the product structure or according to the functions performed by them (e.g. design, verification). Usually, both are done. Nevertheless, this is not sufficient. In traditional hierarchical sub-team approaches, the team members communicate via the sub-team managers (or sub-project managers). With the high level of interaction in HW/SW projects, these people will be a bottleneck and possible source of errors or delays. It is necessary to establish a trustful working link between single engineers in different teams. E.g. a software engineer from one team has to align a hardware interface with an engineer from another team. It is necessary for the sub-project managers to set up these communication links and to check, if exchange of information or code is done. But it does not make sense to try to do the exchange themselves instead of the engineers. This requires a good team building and a good reporting structure. It can help to set up cross-team structures according to major features, including members of different disciplines as hardware, software and concept development, as well as integration engineers who take care of the validation for the integrated HW/SW solution. When setting up the sub-team structure, it is crucial to avoid cutting to many of those links and it makes sense to locate one team in one site to allow informal communication between team members [9].

It is important to let sub teams do their internal planning themselves. This creates a principal-agent-relation and leads to a higher level of commitment by the sub team. Nevertheless, the synchronization dates and deadlines and the synchronization milestones have to be agreed jointly between the overall project manager and the sub teams. Consequently, the overall project manager does his top level planning based on the synchronization events.

B. Joint Core Team

To handle the distributed sub-team structure a joint core team setup is helpful. This core team consists of the sub-project managers and the most important functional lead engineers. It is headed by the overall project manager. The core team is the place to align major deliveries between sub-teams and to decide about the conclusion of important milestones. It is certainly not the place to discuss technical details. The focus is on synchronization of the sub-teams and on the generation of an overall project status and risk report. The core team communicates the status to the respective management and possibly to the customer. This guarantees an aggregated view of the high dynamics within the teams.

Since such a core team can be pretty big (15-20 people) and since it is usually distributed over several sites and time zones, it is necessary to keep the regular meetings short and to have a strict communication discipline. Otherwise, most of the people will waste their time in the meeting while listening to bilateral discussions between few team members. A rule of thumb is to use the meeting only for decisions (e.g. milestones) and for plan updates relevant for all core team members. Technical discussions have to be delegated.

In the example project, core team meetings took place on a weekly basis, due to the need for synchronization. The importance of these meetings for sub-teams lies in the fact that, there, they were able to find out problems or issues that were happening in other sub-teams and may affect them in the future, and also because they were able to expose their own blocking points when management support was needed.

C. Configuration Management

Since the design data is split into a high number of files (several thousands) and since there is a high number of different versions of files (e.g. 10-20 design iterations, even in the range of hundreds for selected modules), it is necessary to apply a sophisticated configuration and data management. Apart from providing the right file versions at the right time to the right people, the configuration management has to ensure the consistency of the design data over the involved sites.

Configuration management has to make sure that the right version of a software component is simulated together with the right hardware component. While integrating different components to a complete system, it has to be ensured that the right versions are chosen to guarantee a fit [10]. Apart from that it has to be considered that different functional teams are working on different versions of the files, e.g. the software engineers use an older but more stable version of the hardware design than used by the hardware engineers for their updates.

D. Synchronisation

During the planning for such a project the most difficult thing is to synchronize the different development tasks. Due to the different development cycle times of the

different functional development teams it is required to optimize the exchange events when input from one team is needed by another team. These synchronization points are the most sensitive events in the project plan. If a delay happens the whole schedule can be affected [3][13].

Therefore, it is necessary to prepare the exchange events well in advance. The delivering team has to understand what to deliver to whom at what time. And the receiving team has to be prepared about what to expect and what to do with that delivery. Therefore, deliveries between teams should be defined in a delivery specification. The delivering team can use this specification to prepare for the delivery. The receiving team can use it to check if they got the right thing. The review of the delivery should be done jointly. By doing this, a principal-agent-relation is established and the teams commit to execute the exchange on time and on quality. In an ideal case each delivery is accompanied by a joint review, yielding a sign-off by the receiving party.

E. Rapid Prototyping

To cope with the high dynamics of project execution and with the need of early co-simulation and co-verification of hardware and software modules it is necessary to release a high number of prototypes of the overall system and sub systems. These prototypes allow the check of high level functions of the system. Nevertheless, the short development cycle times do not allow to wait for simulation results since the simulation time goes up drastically for system level simulations. A solution can be the use of virtual prototypes.

Virtual prototypes allow the early verification of both software and hardware. Based on the virtual prototype, engineers can track bugs in the software design, as well as in the hardware design. The use of virtual prototypes makes it possible to shorten verification times and therefore improves execution times of the whole project.

F. Critical Chain Project Management

Due to the highly interdependent and connected character of the project plan for a large HW/SW codesign project, the application of the traditional critical path methodology does not make sense. The critical path in such a project may change every few days and many other paths are usually not much less critical.

A good approach to get an impression of the overall level of buffer in such a project is the methodology of critical chain project management (CCPM) [6]. This methodology collects the amount of slack in all paths by applying buffers. This gives an impression how critical the whole project is and how much impact can be expected if delays occur.

The use of CCPM was evaluated during the research project, not because it was applied on it, but because its principles were compared to the techniques that were actually used. In the project studied, two levels of management were clearly defined; first the project level,

where the project as a whole was managed through synchronization milestones and second, the sub-project level, where more traditional techniques were used for tracking and controlling.

Synchronization milestones are distributed through the whole project life-cycle and they are essential for consolidating the work developed within different sub-teams. These milestones involve the setting of deadlines: all deliveries that are needed for considering a milestone completed must be ready at a certain date. This is contrary to some of the CCPM principles, which involve not using milestones and deadlines to avoid the Parkinson Law and the student syndrome.

Instead of managing buffer consumption, with the milestones system, the slack consumption is managed. In the project level the float time from sub-projects deliveries to the milestones is monitored. This allows knowing in advance if they could be in danger and act in consequence.

On the other hand, for the subproject level it is important to consider as critical not only the longest path or the one that signal the final date of the sub-project, but all the paths that lead to the sub-project deliveries. Therefore, in this case, sub-project managers have to deal with multiple critical paths at the time.

Besides the advantages that have been attributed to CCPM, such as having one single and not changing critical chain for the whole project, the conclusion is that its implementation, in an ongoing HW/SW codesign project of the dimensions described in this paper, would become a risk for the project. Especially in a case like the one studied where management methods are so well established and understood for the team.

An additional method that could be implemented in order to improve the project tracking and controlling is the Earned Value Analysis (EVA). In the research project, it was observed that budget and time data are managed separately, what makes difficult EVA implementation. Nevertheless, one of the techniques used to monitor the tasks' progress within a sub-project is the calculation of the progress backlog, meaning the difference between the planned work and the actual work in time units. Which is similar to what the EVA calls schedule variance, but in this case the measure is taken in terms of money units.

The combination of the different methods in both, project and subproject levels, is what helps to keep the project on track and to achieve the compromised deadline on time.

IV. CONCLUSION

By introducing the mentioned ideas into large HW/SW codesign projects the execution quality can be increased and the transparency and planning quality is improved. Additional methods like the introduction of schedule risks into the plan by doing Monte Carlo Simulations can help to get a higher planning quality [11].

Research showed that complex methods for tracking and controlling the project execution are needed, but that they need to be planned with anticipation. The complexity of the methods itself is the main reason for this conclusion, because changing from one method to another in an ongoing project requires a big amount of effort and resources and will create a new risk for the project.

REFERENCES

- [1] ASSIMAKOPOULOS, N. A.: Systematic Industrial Management of HW/SW Codesign. *The Journal of High Technology Management Research*, Volume 9, Number 2, pp. 271-284, 1998
- [2] BOEHM, B.W.: Software Engineering Economics. *IEEE Transactions on Software Engineering*, Vol. SE-10, No.1, pp. 4-21, 1984
- [3] CIOFFI, D. F.: Managing Project Integration. *Management Concepts*, Vienna VA, US, 2002.
- [4] DE MICHELLI, G.; ERNST, R.; WOLF, W.: *Readings in hardware/software co-design*. Academic Press, 2002
- [5] GLASS, R.L.: *Software Runaways: Monumental Software Disasters*. Prentice Hall, 1997
- [6] GOLDRATT, E. M.: *Critical Chain*. New York: North River Press, Croton-on-Hudson. 1997
- [7] HARRISON, F.L.; LOCK, D.: *Advanced project management: a structured approach*. Edition 4, Gower Publishing, pp. 70 ff., 2004
- [8] KALING, M.; WOLFF, C.: *Six Principles for Project Integration in Large Projects*. International Research Conference at the University of Applied Sciences in Dortmund, 2010
- [9] KERZNER, H.: *Advanced project management: best practices on implementation*. 2nd Edition, John Wiley a Sons, pp. 461 ff., 2004
- [10] KETTUNEN, P.: Managing embedded software project team knowledge. *IEE Proceedings Software*, volume 150, number 6, pp. 359-366, 2003
- [11] KWAK, Y.H.; INGLALL, L.: Exploring Monte Carlo Simulation Applications for Project Management, *Risk Management*, 9, pp. 44-57, 2007
- [12] LOCK, D.: *Project management*, Edition 9, Gower Publishing, Ltd., pp. 416 ff., 2007
- [13] PMI: *A Guide to the Project Management Body of Knowledge: PMBoK Guide Fourth Edition*. PMI, 2008.
- [14] YOURDON, E.: *Death March*. 2nd Edition, Prentice Hall, 2004